Journal of Industrial Engineering International *Islamic Azad University, Tehran South Branch*
September 2005, Vol. 1, No. 1, 10 - 19

An ACO algorithm for one-dimensional cutting stock problem

K. Eshghi

Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran

H. Javanshir*

Islamic Azad University, Science and Research Branch, Tehran, Iran

Abstract

The one-dimensional cutting stock problem, has so many applications in lots of industrial processes and during the past few years has attracted so many researchers' attention all over the world. In this paper a meta-heuristic method based on ACO is presented to solve this problem.

In this algorithm, based on designed probabilistic laws, artificial ants do select various cuts and then select the best patterns. Also because of the problem framework, effective improvements has been made to problem solving process. The results of that algorithm in sample problems, show high efficiency of the algorithm in different levels of problems.

Keywords: One-dimensional cutting stock problem; Ant colony optimization; Trim loss

1. Introduction

The one-dimensional cutting stock problem, that in this paper is called "one-dimensional cutting problem" has so many applications in lots of industrial processes [5,8,9] and during the past few years has attracted an increasing attention of researchers all over the world [1,12]. This attention has been mostly focused on the solution to the problem in cases with the stock of the same length or with a few different standard lengths.

Most standard problems related to one-dimensional cutting problem are known as NP-complete. However, in many cases these kinds of problems can be modeled by means of mathematical programming and a solution can be found by using approximate methods and heuristics. The objective is to design a plan of one-dimensional cutting of a certain number of pieces of same lengths (stock lengths), into a large number of short pieces (order lengths), which will minimize the overall trim loss considering different conditions that may appear in practice.

Using Dyckhoff's typology [4], the one-dimensional cutting problem with enough required material available can be described as:

1. Dimensionality
   N) Number of dimensions:
2. Kind of assignment:
   B) All large objects and a selection of small items
   V) A selection of large objects and all small items
3. Assortment of large objects:
   O) One large object
   I) Many identical large objects
   D) Different large objects
4. Assortment of small items:
   F) Few items of different dimensions
   M) Many items of numerous different dimensions
   R) Many items of relatively few dimensions
   C) Many identical items

In this paper, the *1/V/O/R* has been considered, where 1 refers to one-dimensional problem, *V* means that all items must be produced from a selection of large objects, *O* means that one large object and *R* indicates many items of relatively few dimensions. The algorithm presented in this paper, also could be used for *1/V/I/R*, where I means that Many identical large objects.

Dyckhoff classifies the solution of one-dimensional cutting problems into two groups: *item-oriented* and *pattern-oriented* approach. *Item-oriented* approach is characterized by individual

*Corresponding author. E-mail:hjavanshir@yahoo.com treatment of every item to be cut. In the *pattern-oriented* approach, at first, order lengths are combined into cutting patterns, for which - in a succeeding step - the cutting frequencies are determined that are necessary to satisfy the demands. The constraints in the pattern-oriented approach are based on the algorithm that Gilmore and Gomory have developed [6,7]. However, a pattern-oriented approach is possible only when the stock is of the same length or of several standard lengths, and an item-oriented approach is used when all stock lengths are different and frequencies cannot be determined. The authors selected *pattern-oriented* approach for solving the cutting problem.

2. Describing one-dimensional cutting problem model

In so many industries, the cost of raw materials is the most percentage of the total cost (sometimes more than 80%). Then lots of attempts have been done to increase materials utilization. The cutting problem is one of the well-known operation research problems that is defined to make better use of materials. In general, the cutting problem could be defined as:

One or more large objects are available and we want to make some small required items by cutting them. In this problem the cutting method should be determined in a way that minimum trim loss is made or smaller objects are cut, and used.

The cutting problem was firstly described in 1939 by Kantorovich for one-dimensional cutting [10]. In 1960s P.C. Gilmore and R.E. Gomory published four famous papers about one and two-dimensional cutting problems. Their first paper was published about the application of linear programming in solving one-dimensional cutting problems in 1961 and it was a real start for representing techniques used in actual problems. Publishing Gilmore and Gomory papers caused a new movement in analyzing and solving the cutting problem and most papers published about cutting problems till now have referenced Gilmore and Gomory papers. While expanding the application of computer in operation research problems and developing techniques and methods of modeling and problem solving, the cutting problem has been developed.

2.1. One-dimensional cutting problem

The one-dimensional cutting problem could be described as:

There are some large objects and we want to cut them (for ordered items that have two identical dimensions) in a way that minimum trim loss is made. Cutting problem dimension is the degree of freedom for decision making. If the two dimensions of ordered items and used large objects are the same, only decision for the way of cutting third dimension should be made and therefore, cutting process has one dimension.

The main objective in cutting problems is decreasing the cost of losses. Meaning that we want cutting patterns, and number of using them to make minimum trim loss cost. Of course, in some cases, time and cost of set up change of a cutting machine is considerable [14].

Represented models for cutting problems are affected firstly by kind of its hypotheses and secondly by the modeling method.

2.2. Defining model

The problem model is as follows:

$$\text{Min } X_0 = \sum_{j=1}^{m} S_j X_j + \sum_{i=1}^{n} V_i \qquad (1)$$

Subject to:

$$\sum_{j=1}^{m} O_{ij} X_j \geq D_i \qquad i = 1,\ldots,n \qquad (2)$$

$$X_j \geq 0, \text{ Integer} \qquad j = 1,\ldots,m \qquad (3)$$

where:

X_j : Number of large objects that have been cut by *j* th cutting pattern.
S_j : Amount of loss in pattern *j* th.
O_{ij} : Number of *i* th item cut by pattern *j* th.
D_i : Number of demond for *i* th item.
V_i : Surplus production of *i* th item (surplus variable of model constraints) that is calculated as follows:

$$\sum_{j=1}^{m} O_{ij} X_j - V_i = D_i$$

$$\Rightarrow V_i = \sum_{j=1}^{m} O_{ij} X_j - D_i \ , \quad i = 1,\ldots,n \qquad (4)$$

m : Number of effective cutting patterns.
n : Number of different required items.

If L is the length of large objects and ℓ_i is the length of i th item, it is clear that we will have:

$$\sum_{i=1}^{n} O_{ij}\ell_i + S_j = L \quad , \ j = 1,\ldots,m \qquad (5)$$

It is worth mentioning that it is possible to write objective function as Min $X_0 = \sum_{j=1}^{m} X_j$.

3. Ways to solving one-dimensional cutting problems

The one-dimensional cutting problem is one the NP-complete problems and therefore many different ways are represented to solve it. These represented methods could be divided into two main groups: optimization methods and heuristics. Among optimization methods, some algorithms based on dynamic programming and methods based on linear programming could be mentioned. Heuristics are divided into different types like metaheuristic algorithms.

3.1. Metaheuristic algorithms

Complexity in current problems made the optimization methods not be able to gain global optimum or use lots of time to reach this answer. These problems usually because of their own reasons have many local optimum and only using current optimization methods is so expensive and sometimes impossible. Therefore metaheuristics are represented that some of them are: Tabu Search (TS) algorithm, Genetic Algorithm (GA), Simulated Annealing (SA) algorithm and Ant Colony Optimization (ACO). Nowadays these methods because of not being designed for a particular problem and reaching to the answer in the minimum possible time, have attracted an increasing attention of researchers.

3.2. Aco metaheuristic algorithm

Metaheuristic optimization algorithm based on ant's behavior (ACO) was represented in the early 1990s by M. Dorigo, V. Maniezzo and A. Colorni [2,3]. This algorithm is inspired of ant's social behavior. Ants have no sight and could find shortest way from food to their nest by chemical materials called Pheromone that they leave when moving [3].

When ants are walking they leave Pheromone and follow (catastrophically) other ants' left Pheromones. Ants like the way that has the most amount of Pheromones. The process of finding the shortest way by using Pheromone is shown in figure 1.

Look at figure 1-a. Ants have arrived at a junction and should decide to go upward or direct. In this time there is no memory about selecting the best way. So ants choose their way randomly. It is estimated to see that one half of ants will go upward and others will go in the direct way that is shown in figure 1-b. Because the direct way is shorter and by supposing that ants' walking speed are identical, much more ants could pass this way per unit time and therefore it makes this way filled with Pheromone move quickly. By the time, Pheromone difference between two ways increases and after sometime Pheromone difference between two ways becomes enough great to affect the ants' way selection. This is shown in figure 1-c. When ants are coming back because of finding more Pheromones in the down way, will probabilistically prefer this way. This process will continue with a positive feedback, meaning that increasing the number of selecting this way causes increasing Pheromone and increasing Pheromone causes increasing the number of selecting this way. After sometime all ants will choose shorter way to continue the movement.

First ant colony optimization algorithm was based on this ants' behavior. ACO algorithm was firstly used in solving traveling salesman problems (TSP) and then was used in solving combinational optimization problems that among them we could indicate to quadratic assignment problems (QAP), routing problems, graph coloring problems, etc. In the most problems that have been solved by ACO algorithm, results indicate superiority of this method to other metaheuristics.

4. Representing an ACO algorithm for solving one-dimensional cutting problem

In this section, an algorithm based on ACO method is represented for solving one-dimensional cutting problem. In this kind of algorithm, we should use artificial ants which their maximum number is total required cuts. Firstly every artificial ant selects a stochastic probability rule to choose cut (item) type and then selects desired pattern to perform that cut by

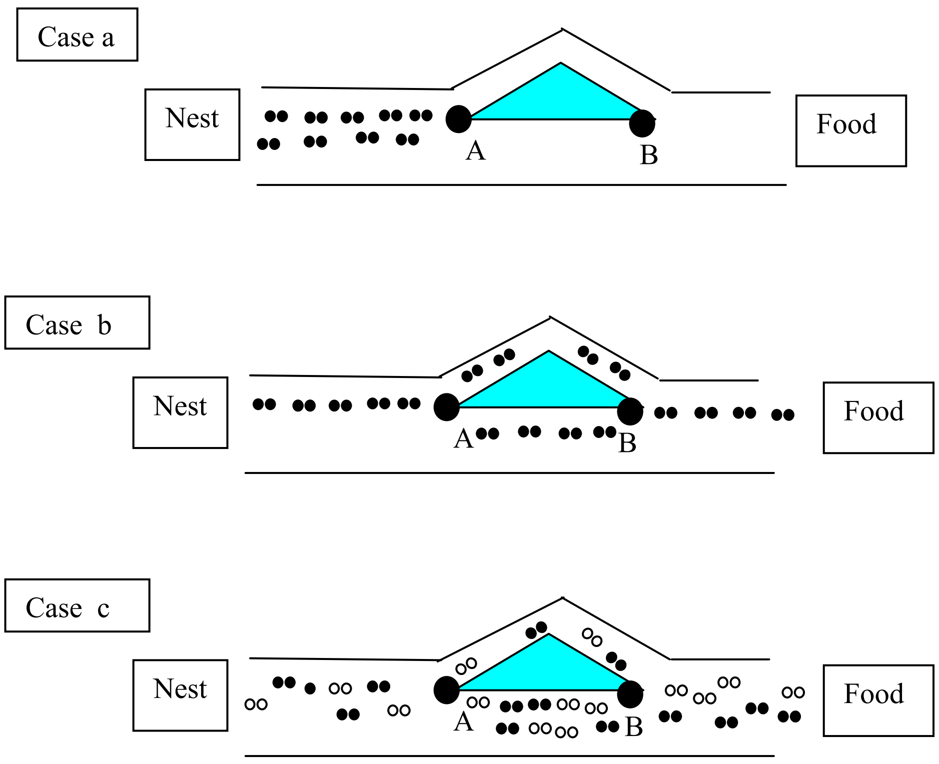

Figure 1. The process of finding shortest way between two points by ants.

another probabilistic rule. Finally after updating the amount of required left cuts and because of producing that pattern, selecting new cuts and patterns by other ants using the remaining information of others is performed till there is no cutting needs. Also after some program iteration, Pheromone evaporation is done to prevent repeated similar answers, and to escape local optimum.

4.1. The method of efficient cutting patterns generation

Because in every iteration of algorithm, there is a need of using cutting patterns by ants, in order to do this we do as follows:

In order to generate efficient cutting patterns that no one has superiority to the others, we could use different methods like Pierce in 1964 and Suliman in 2001 that were used to generate all effective cutting patterns for large objects [11,13,15].

In Pierce method, if L is the length of the large objects and ℓ_1 , ℓ_2 , … , ℓ_n are the lengths of the items needed, in a descending length order and M is the maximum trim loss allowable, then we have:

*Step 1.*Set

$$\alpha_1 = \left[\frac{L}{\ell_1}\right],$$

$$\alpha_2 = \left[\frac{L-\alpha_1\ell_1}{\ell_2}\right], \dots, \alpha_n = \left[\frac{L-\sum_{i=1}^{n-1}\alpha_i\ell_i}{\ell_n}\right],$$

where $[g]$ is the largest integer less than or equal to g.

Step 2. If $L-\sum_{i=1}^{n}\alpha_i\ell_i \le M$, then $\begin{bmatrix}\alpha_1\\ \alpha_2\\ \vdots\\ \alpha_n\end{bmatrix}$ is an efficient cutting pattern.

Step 3. If there is an i , $1 \le i \le n\text{-}1$, such that $\alpha_i > 0$, then let j be the largest such i and go to step 4. If not, terminate the procedure. All efficient cutting patterns have been identified.

Step 4. Set

$$\alpha_j = \alpha_j - 1 ,$$

$$\alpha_{j+1} = \left[\frac{L - \sum_{i=1}^{j} \alpha_i \ell_i}{\ell_{j+1}}\right], \ldots, \alpha_n = \left[\frac{L - \sum_{i=1}^{n-1} \alpha_i \ell_i}{\ell_n}\right],$$

Go to step 2.

Step 5. If there are two pattern $\begin{bmatrix} O_{1k} \\ O_{2k} \\ \vdots \\ O_{nk} \end{bmatrix}$, $\begin{bmatrix} O_{1j} \\ O_{2j} \\ \vdots \\ O_{nj} \end{bmatrix}$ that for all *i* we have $O_{ik} \le O_{ij}$, then the cutting pattern $\begin{bmatrix} O_{1k} \\ O_{2k} \\ \vdots \\ O_{nk} \end{bmatrix}$ because of the other pattern's superiority should not be considered.

After generating effective cutting patterns, it is possible to decrease required number of items by following this note that:

If a kind of item is only produced by one pattern, then firstly it will be produced in required numbers. Since every pattern could make different cuts, then whole number of total cuts is calculated and left requirement of every kind of cut and problem modeling information is updated.

4.2. Probabilistic rules and selecting cut and pattern

Now, ACO algorithm is described for solving one-dimensional cutting problems. It is necessary to remember that in the ACO designed for solving one-dimensional cutting problem, artificial ants are used less than the number of total required cuts. So in this algorithm, the number of artificial ants in every iteration is a variable depending on total required cuts. In every iteration firstly every ant using a probabilistic rule, selects it is desired cut. For creating this probabilistic rule assume that:

P_i : Number of production of the cut (item) of the type *i* till this time of solving the model (calculated on the number of selected patterns by other ants).

D_i : Whole required demand for the cut (item) of the type *i* .

Then $D_i - P_i$ indicates left required demand for the cut of the type *i* based on the produced items till now, that we show by M_i. Then it is possible to write:

$$M_i = \begin{cases} D_i - P_i & \text{if } D_i - P_i > 0 \\ 0 & \text{if } D_i - P_i \le 0 \end{cases} \tag{6}$$

If the sum of total left demands is shown with *TM*, then:

$$TM = \sum_{\forall i} M_i \tag{7}$$

Note that if *TM=0*, there is no need to choose another type of cut by any other ant and therefore no pattern is selected in this iteration and the iteration is over. So the probability of selecting every kind of cutting in a moment by every ant equals:

Probability of selecting the cutting type *i* by any ant $= \dfrac{M_i}{\sum_{\forall i} M_i} = \dfrac{M_i}{TM}$ (8-1)

Obviously this probability for every kind of cutting that has no demands is equal to zero. Also cuts having more required left, has more probability by itself.

That probabilistic rule is improvable as follows:

Cuts (and patterns) selection by ants in the last iteration, give important information including surplus production of every type of cutting in the last iteration.

Assume that:

P_i' : Number of whole production of type *i* in the last iteration (calculated on the number of selected patterns by ants in the last iteration).

ℓ_i : The length of demanded the cut of type *i* .

V_i' : Amount of surplus cut production of type *i* in last iteration.

Now this could be written:

$$V_i' = (P_i' - D_i) * \ell_i \tag{9}$$

If sum of the total different surplus cuts productions indicating with TOV' equals zero, used probabilistic in the last iteration are suitable and again are assumed in this iteration, else we should do the following:

So in particular cases we do not want cuts without surplus production in the last iteration have very great probabilities and cuts with high surplus production (and sometimes only cut with surplus production) have little probabilities, then we equal amount of surplus production of cuts without surplus production to the small percentage of minimum of other different cuts (except zero) surplus production that are selected to be equal to equation (9). When performing algorithm on the case studies, this amount is equal to 5%.

Now recalculate the sum of cuts with more surplus production in the last iteration shown with TOV' and find improvement probability coefficient resulted equation (8-1) in this way:

Z_i : Improvement probability coefficient for selecting the cut of type *i* by artificial ant.

$$Z_i = 1 - \frac{V_i'}{TOV'} \tag{10-1}$$

Since we want cuts with more surplus production in the last iteration to be less selecting probability by ants to make these cuts have less surplus production in this iteration.

In this case, equation (8-1) is improved as follows:

$$\text{Probability of selecting the cutting type } i \text{ by any ant} = \frac{M_i}{TM} * Z_i \tag{8-2}$$

The important point is that in using the rule above, the normalized form of probabilities should be applied so that sum of the resulted probabilities equals to one.

Now by using point below, equation (10-1) will be improved:

As seen V_i'/TOV' indicates percentage of surplus cut of type *i* production in the previous iteration. But this proportion is so important and could improve the process of program performing iterations and its proportional importance could be strengthened if after some sequential iterations (for example 10 iterations) there is no improvement in obtained answers. Assume that importance coefficient is shown with α , then firstly α=1, but if after some sequential iterations (for example 10 iterations) there is no improvement in the problem solving process, parameter α misses its value by a defined amount (for example 10%). So the ratio V_i'/TOV' can be strengthened of the same amount. Or firstly the value of parameter α is selected less than one and proportional importance of the equation above should be strengthened from the beginning. So in equation (10-1), improvement probability coefficient for selecting cut of type *i* will be changed as follows:

$$Z_i = 1 - \left(\frac{V_i'}{TOV'}\right)^{\alpha} \quad , \quad 0 < \alpha \le 1 \tag{10-2}$$

It is worth mentioning that in the first program iteration, all Z_i are assumed equaled to one.

Immediately after selecting the cut of type *i* by every ant, pattern of type *j* including the cut of type *i* should be selected by the same ant. Therefore patterns with the cut of type *i* are considered. Assume that:

O_{ij} : Number of obtained units from the cut of type *i* in every selected pattern of type *j*.

N_j' : Number of selections by ants from pattern of type *j* in the previous algorithm iteration.

S_j : Amount of loss in pattern *j* th.

Then $N_j'.S_j$ indicates total loss resulted from selecting pattern of type *j* in the previous algorithm iteration. Afterwards the probability of selecting pattern *j* for cut of type *i* by artificial ant $(P_{(j)}')$ is calculated as follows:

$$P_{(j)}' = \frac{1}{N_j'.S_j}.Z_j' \tag{11-1}$$

$$Z_j' = O_{ij} \tag{12-1}$$

Here Z_j' is the improvement probability coefficient for selecting pattern of type *j* that indicates the number (weight) of the desired cut of type *i* in the pattern *j*. Of course it is necessary to normalize probabilities above $(P_{(j)})$. It means:

$$P_{(j)} = \frac{P_{(j)}'}{\sum_{\forall j \in \Omega i} P_{(j)}'} \quad , \quad \Omega_i = \{ j \mid \forall O_{ij} > 0 \text{ , for desired cut of type } i \} \tag{13}$$

The important point is that the denominator of the equation (11-1) should not become zero. So for patterns that had no selection in the previous iteration or its amount of wastage was zero, then we assume $N_j'.S_j$ small percent of the minimum $N_j'.S_j$ of other patterns (except zero) having the cut of type *i*.

For improving equation (11-1) the following method could be used:

Again consider equation (6). If in a moment of solving, M_i for every cut of type *i* becomes zero, then the length of that cut type will be considered as waste. So in the equation (11-1) the value of S_j in this moment starts to increase as following that could make less production of the cut of type *i*.

$$S_j = S_j + \sum_{\forall i \in \delta_i} O_{ij}.\ell_i \quad , \quad \delta_i = \{i \mid \forall M_i = 0 \ , \text{ in the desired pattern } j\} \qquad (14)$$

On the other hand, probability improvement coefficient Z'_j is also improvable for selecting pattern of type *j* that is improved in (12-1). Because by selecting the desired cut *i* by every ant, we want to have selected pattern of type *j*, strengthen more production of this type of cut. So it can be written:

$$Z'_j = (O_{ij})^{\beta} \quad , \quad \beta \geq 1 \qquad (12\text{-}2)$$

Also equation (11-1) can be improved as follows:

$$P'_{(j)} = \left(\frac{1}{N'_j.S_j}\right)^{\gamma}.Z'_j \quad , \quad \gamma \geq 1 \qquad (11\text{-}2)$$

In those above equations, parameters β and γ are best pattern selecting probability improvement parameters.

When performing above equations in the sample problems, values of the parameters β and γ are considered as β=1 and $1 \leq \gamma \leq 2.5$.

It is necessary to indicate that if for a cut type *i* , all S_j were equal to zero, then equation (11-2) for that cut becomes as below:

$$P'_{(j)} = \frac{O_{ij}}{\sum_{\forall i} O_{ij}} \qquad (15)$$

Also in the first algorithm iteration, all N'_j and even all Z'_j could be considered equal to one. And also it is possible to use the following equation by the normalized probabilities form instead of (11-2):

$$P'_{(j)} = 1 - \frac{S_j}{\sum_{\forall j \in \Omega i} S_j} \qquad (16)$$

Making various answers process in this algorithm is in this form that in every time using equation (11-2), it inverts the movement direction from answers. Or if in one time using it a worse solution compared to the last iteration is made, then the next use, catches better solution of the problem. This means that if the direction movement toward optimum solution requires improvement, the search continues by sequential directions inverting. So the solution algorithm does not stop in a local optimum. Doing this work is equivalent to the meaning of Pheromone evaporation in the ACO algorithm.

After obtaining a basic feasible solution, equation (11-2) will affect and shows movement direction toward optimum solution. Now every time that the algorithm solution in one iteration is better than the previous iteration, or in other words movement direction is recognized toward improvement, then instead of equation (11-2) we use the following equation:

$$P'_{(j)} = \frac{N'_j}{S_j}.Z'_j \quad , \quad Z'_j = O_{ij} \qquad (17\text{-}1)$$

It is worth mentioning that equations (12-2), (13) and (14) could be completely applied like last times and equation (17-1) can be improved in this way:

$$P'_{(j)} = \frac{N'_j}{(S_j)^{\gamma}}.Z'_j \quad , \quad Z'_j = (O_{ij})^{\beta} \qquad (17\text{-}2)$$

After selecting the pattern by every ant, because every pattern could produce various cuts, number of products from the cut type *i* till this time will be updated according to equation (18):

$$P_i = P_i + O_{ij} \quad , \quad \forall i \qquad (18)$$

Of course, the above operations will be repeated by so many artificial ants until all needs are met (means that $TM = 0$). Thus at the end of algorithm, we will have a desired feasible solution.

Now by using this program iteration information, repeat the algorithm for some times until the best solution is obtained. Parameters α, β and γ make necessary improvement in program iteration.

In order to improve obtained answers from every program iteration finally we could behave as follows:

After finding the best answer between obtained answers, the number of selected patterns can be decreased well that amount of surplus production from patterns will be minimum.

It is worth mentioning that, this operation can be done in every program iteration after recording needed information of every iteration only for finding the best answer in different program iterations and then finding the best obtained answer till the time that is recognized and saved. The stoppage condition of the algorithm is that if after some iterations (for example 20 iterations) there is no change in the best obtained answer, then the algorithm is stopped.

4.3. Summary of ACO algorithm for problem solving

The ACO algorithm could be summarily changed to psedu-code for solving a one-dimensional cutting problem as follows:

```
Procedure ACO_Cutting_Stock
  Data Entry: L , $\ell_i$ , $D_i$ ; i = 1,…,n
  Patterns Generated by Pierce or Suliman Method: $O_{ij}$ , $S_j$ ; j = 1,…,m
  Find Initial Feasible Solution by Equs. (8-2)&(16)
  Calculate $P'_i$ , $V'_i$ , $N'_j$ , $X_{0(k=1)}$ for Initial Feasible Solution
  Best-Solution = $X_{0(k=1)}$
  Repeat: (For k + 1)
     Foreach Ant: Until TM = 0
        Calculate $P_i$ , $M_i$ , TM by Equs. (6),(7),(18)
        Select Cut ( i ) by Equ. (8-2)
        If ( $X_{0(k)} \leq X_{0(k-1)}$ ) Then
           Select Pattern ( j ) by Equ. (17-2)
        Else
           Select Pattern ( j ) by Equ. (11-2)
        End-If
     End-Foreach
     Calculate $P'_i$ , $V'_i$ , $N'_j$ , $X_{0(k+1)}$
     Improve $V'_i$ , $P'_i$ , $N'_j$ , Calc. Again $X_{0(k+1)}$ & Best-Solution
     Check Stopping Criteria
  End-Repeat
End-procedure
```

5. Computational results of the model solving method

In order to examine the provided algorithm in this paper, a software using *Visual Basic 6.0* is supplied that according to the mentioned algorithm, solves sample problems. Then the algorithm software for sample problems has been performed using a PC with characteristics of pentium 4 with 2.8 GHz processor and 512 MB RAM.

In table (1) computational results obtained from average solution of 18 group of sample problems that have different dimensions, are provided. In every group of problem levels TP1 to TP10 ten different sample problems and for problem levels TP11 to TP18 five different samples were produced, and obtained results are gained considering to the average results in these samples. So totally 140 problems are produced and solved. Sample problems dimensions are so considered that we are able to compare final algorithm answer with an answer using column generation method for solving a one-dimensional cutting problem.

The method of producing sample problems in every group is that for entering software, firstly the length of large object is produced randomly an integer number between [20,100] and then by considering the number of different needed items (n), the length of the demanded items are integer random numbers that are produced using large object length and their demands are random integer numbers between [50,400]. Then program according to above entrances generates efficient cutting patterns in needed numbers (m) using *Pierce* method. Now by considering the number of produced patterns, parameters α , β and γ are determined by the user and program will continue until getting the final answer.

In the columns of table (1) from left, name of sample problems (TP), number of required various items (n), number of generated patterns by software (m) that are classified, the best values of parameters α , β and γ in different program performings, difference between obtained result by algorithm and optimum solution of the problem (Gap-Opt) and average needed CPU working time (when performing program) to get the final problem answer measured by the dimension of second (Time) are shown.

According to computational experiences, the software of the provided algorithm in this paper can find optimum solution of all problems of different levels with little solving time that this indicates the algorithm strength. In other words, the optimum solution in all 140 sample problems is obtained.

Also studying the obtained results from sample problems shows that by enlarging parameter γ, problem desired answers tend to select less different number of patterns. With due attention to solved sample problems, γ=2 is a good value for the above solving method. By reducing parameter α, there is

Table 1. Computational results obtained of 18 groups of sample problems.

TP	n	m	α	β	γ	Gap-Opt	Time
TP01	3	$6 \le m \le 4$	1	1	1	0	1.5
TP02	3	$15 \le m \le 7$	1	1	1	0	2
TP03	4	$15 \le m \le 8$	1	1	1	0	4
TP04	4	$30 \le m \le 16$	1	1	1	0	4.5
TP05	5	$20 \le m \le 8$	1	1	1	0	6
TP06	5	$40 \le m \le 21$	1	1	1	0	15
TP07	5	$70 \le m \le 41$	1	1	1	0	18
TP08	6	$30 \le m \le 10$	1	1	1	0	11
TP09	6	$60 \le m \le 31$	0.5	1	1	0	40
TP10	6	$110 \le m \le 61$	1	1	1	0	43
TP11	7	$70 \le m \le 25$	1	1	1.5	0	56
TP12	7	$150 \le m \le 71$	1	1	1.5	0	65
TP13	8	$80 \le m \le 30$	1	1	1.5	0	61
TP14	8	$165 \le m \le 81$	1	1	2	0	81
TP15	9	$90 \le m \le 30$	0.5	1	1.5	0	84
TP16	9	$170 \le m \le 91$	0.5	1	2	0	92
TP17	10	$90 \le m \le 30$	1	1	1.5	0	119
TP18	10	$180 \le m \le 91$	1	1	2	0	146

more equilibrium between surplus productions of items. $\alpha = 0.5$ makes a suitable equilibrium to the optimum solution in the sample problems.

Parameter β is a complement for parameter γ . In the sample problems because by putting $1 \le \gamma \le 2.5$ we have reached the optimum solution then putting $\beta = 1$ is desirable.

What appears in solving sample problems is that items with shorter length that are produced in more number of patterns and also items with small demands usually have more surplus production.

Despite in large problems like TP18 having 10 kinds of different items with more than 90 patterns, algorithm performance time that resulted in optimum solution, was lower than 150 seconds. Also average algorithm performance time in 140 sample problems in 35.5 second indicating algorithm high speed to reach the optimum solution for the studied sample problems.

6. Conclusions

In this paper, an ACO algorithm is designed for solving one-dimensional cutting problem. In this algorithm based on designed probabilistic laws and improvements, the way of cutting large objects to satisfy demand by artificial ants is provided. By looking at the desired obtained results from performing algorithm, it clears that metaheuristics for solving one-dimensional cutting problem will result in very desired answers with little solving time that could be a good basis for researchers' future researches.

References

[1] Bishoff, B. B. and Waesher, G., 1995, Cutting and packing. *European Journal of Operational Research*, 84, 503-505.

[2] Dorigo, M. and Di Caro, G., 1999, *Ant Colony Optimization: A New Meta-heuristic.* Proceedings of the 1999 Congress on Evolutionary Computation.

[3] Dorigo, M., Maniezzo, V. and Colorni, A., 1999, *Positive Feed back as a Search Strategy.* Technical Report, University of Milan, Italy, 91-106.

[4] Dyckhoff, H., A typology of cutting and packing problems. *European Journal of Operational Research*, 44, 145-159.

[5] Ferreira, J. S., Neves, M. A. and Fonseca, P., 1990, A two-phase roll cutting problem. *European Journal of Operational Research*, 44, 185-196.

[6] Gilmore, P. C. and Gomory, R. E., 1961, A linear programming approach to the cutting stock problem. part I, *Operations Research*, 9, 849-859.

[7] Gilmore, P. C. and Gomory, R. E., 1963, A linear programming approach to the cutting stock problem, part II, *Operations Research*, 11, 863-888.

[8] Haessler, R. W. and Sweeney, P. E., 1991, Cutting stock problems and solution procedures. *European Journal of Operational Research*, 54, 141-150.

[9] Haessler, R. W. and Vonderembse, M. A., 1979, A procedure for solving the master slab cutting stock problem in the steel industry. *AIIE Transactions*, 11, 160-165.

[10] Madsen, Olib B. G., 1988, An application of traveling-salesman routines to solve pattern-allocation problems in the glass industry. *Journal of Operational Research Society,* 39(3), 249-256.

[11] Pierce, J. F. Jr., 1964, *Some Large-Scale Production Scheduling Problems in the Paper Industry*. Prentice-Hall, Englewood Cliffs, NJ. (1964).

[12] Rietz, J., Scheithauer, g., and Terno, J., 2002, Families of non-IRUP instances of the one-dimensional cutting stock problem. *Discrete Applied Mathematics*, 121, 229-245.

[13] Suliman, Saad M. A., 2001, Pattern generating procedure for the cutting stock problem, *International Journal of Production Economics*, 74, 293-301.

[14] Umetani, S., Yagiura, M. and Ibaraki, T., 2003, One-dimensional cutting stock problem to minimize the number of different patterns. *European Journal of Operational Research,* 146, 388-402.

[15] Wagner, B. J., 1999, A genetic algorithm solution for one-dimensional bundled stock cutting. *European Journal of Operational Research*, 117, 368-381.